THE UNIVERSITY OF MICHIGAN

Technical Report 33

CONCOMP

October 1970

CAMA (COMPUTER-AIDED MATHEMATICAL ANALYSIS): A GENERAL DESCRIPTION

Louis W. Wolf

BLANK PAGES
IN THIS
DOCUMENT
WERE NOT
FILMED

T H E U N I V E R S I T Y O F M I C H I G A N

Memorandum 33

CAMA (COMPUTER-AIDED MATHEMATICAL ANALYSIS): A GENERAL DESCRIPTION

by

Louis W. Wolf

CONCOMP: Research in Conversational Use of Computers
ORA Project 07449
F. H. Westervelt, Director

supported by:

ADVANCED RESEARCH PROJECTS AGENCY
DEPARTMENT OF DEFENSE
WASHINGTON, D.C.

CONTRACT NO. DA-49-083 OSA-3050
ARPA ORDER NO. 716

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

October 1970

ABSTRACT

This paper describes CAMA (Computer-Aided Mathematical Analysis), which is a system that attempts to handle all conceivable mathematical operations, from simple numeric procedures to complex algebraic or set-theoretic manipulations. CAMA operates in a timesharing environment with a large central computer and a remote terminal computer with graphical devices.

Some of the features of CAMA are: the input and output of mathematical symbols in their usual mathematical representation; the ability to manipulate these symbols according to predefined algebraic laws; the definition by the user of other algebraic laws; the definition of symbols with user-prescribed meanings; the automatic translation of expressions or equations into any of several languages; and numerical calculations. The user can invoke any of these tasks through the graphical terminal.

TABLE OF CONTENTS

Abstract. iii

Preface ix

1. Summary 1

2. Introduction. 3

3. Hardware Configuration. 9

4. Central Computer Software 11

5. Data Structures 12

6. Task Queue. 15

7. Display File Procedures 16

8. Drawing Capability. 17

9. Dynamic Loader. 18

10. Translators 19

11. Algebraic Manipulation. 22

12. Terminal Procedures 24

Acknowledgments 26

References. 27

LIST OF FIGURES

Figure 1. Hardware Configuration. 10

Figure 2. Data Storage Pack Structure 13

PREFACE

The present (September 1970) state of the CAMA system is not exactly as reported herein. At the end of the CONCOMP Project (August 1970), a number of parts of the system were unfinished or in a non-working condition. The interpreter and the macro processor are in a good working state, although they still have a number of elusive bugs. The nature of the CAMA system requires that various parts of the program get and release space as a matter of course. This practice causes the internal evidence of some of the bugs to disappear before we are able to look for them. However, this difficulty does not prevent the effective use of the system; it makes it only slightly annoying to work with.

The editor is in and working well, along with a full range of other operations which aid the user in constructing and debugging programs. All of the data structure functions are in good condition, as is the tasking operation in the central computer.

During the evolution of CAMA, all of the programs in the terminal computer, including such basic ones as the data structure, DF routines, and RAMP, have undergone various changes in structure and function, and in their calling sequences and return codes. Consequently, a number of system components that were functioning

previously are now in a non-functioning condition. These include the symbol-defining capability, plotting routines, the drawing capability, and several other minor components.

Several parts of the system are not finished or are in only a partially working condition. These include the symbol manipulating capability and the parsing feature. These features can be used by their coders but not by the general user at this time. The application of the symbol manipulation capability to various problems awaits a more usable functioning of the system as a whole.

Some parts of the system function more slowly than is desirable for general use. This slow action is largely due to two factors. A large number of checking procedures are now included in the programs to help detect errors and to aid in debugging. These will be removed as the use proves out various parts of the system. The second most important factor is that many operations that could be delegated to dwell-time tasks are at this time done in sequence with other tasks. As the use of the system picks up these task will be reassigned.

Even though the CAMA system as a whole is not quite complete, in its present state it can be used to great advantage by the average user. Those of us who designed and built CAMA have used it to build new sections of the system and to extend its features, and in so doing have found that the general concept and viability of the system were clearly demonstrated.

1. SUMMARY

This paper describes CAMA (Computer-Aided Mathematical Analysis), a system that attempts to handle all conceivable mathematical operations, from simple numeric procedures to complex algebraic or set-theoretic manipulations. CAMA is not a language, although languages are certainly involved in it. Rather, CAMA provides the user--mathematician, engineer, or scientist--having mathematical manipulations to perform with a more congenial computing environment. CAMA functions in a timesharing system, with a large central computer and a remote terminal computer with graphical devices. This paper describes the overall objectives and modes of operation of the CAMA system; the technical details appear in other papers and reports.[6, 12-14,22]

Some of the features of CAMA are: the input and output of mathematical symbols in their usual mathematical representation; the ability to manipulate these symbols according to predefined algebraic laws; the definition by the user of other algebraic laws; the definition of symbols with user-prescribed meaning; the automatic translation of expressions or equations into any of these tasks through the graphical terminal.

CAMA is efficient in that only those parts which are in immediate use need be kept in virtual memory of

the central computer, while a large file of other parts are available on readily accessible and less expensive disk files. Numerical output can be in the form of displayed graphs, or in more usual tabular form. Hard copy of any graph, equation, or symbolic pattern can be obtained immediately, or at some later time at lower cost.

2. INTRODUCTION

The introduction of large timesharing computers, as well as small, fast, inexpensive computers, and graphical devices to support them, has made possible many applications of the computer which were at best difficult in the past.[1,2] Any mathematician, engineer, or scientist who has had to handle mathematical manipulations, turn these into code, debug that code, and plot numerical results, knows what tedious chores these are. With new hardware, however, the mathematician or engineer can now communicate with the computer in more familiar terms, that is, with the same mathematical symbols that he uses normally on a sheet of paper. The user can draw symbols, which in turn communicate information to the digital computer. Procedures for mathematical manipulation of these symbols can be invoked and the results transmitted back to the user in the form of the symbols he has defined, and not in the form of obscure mnemonics. CAMA was conceived not as a language for performing these operations, but as a group of aids to assist the mathematician in whatever operations he wishes to perform. These aids include such things as graphical input of symbols and expressions, symbol manipulation, translation of these symbols into numerical processes, and automatic display of the results.

The designers could not, of course, anticipate all the possible uses of CAMA, so many features were purposely left open to enable users to define their own operations. There are many predefined operations; however the knowledgeable user will be able to augment the standard operations in as much as CAMA is flexible and adaptable. He may modify the commands, modify the data storage, modify the syntax of the languages, modify the meaning of words, modify the meanings of operators. He may also define new symbols and attach new meanings to old symbols.

A CAMA design objective is ease of use. A novice, without any experience in programming, should be able to use many of the operations without difficulty. For example, a novice can sit down at the console and do the following operations: select a group of symbols to use in his equations, say an ordinary differential equation; construct a differential equation on the display screen, define the boundary conditions, and select a method of solution, for example, Runge-Kutta. He may then select the mode of the output: a tabulation of the results of the solution and its derivatives, and/or a graphical display of the solution. He may then cause the differential equation to be solved and receive the results. If he then wishes to solve the differential equation by a different method, he selects that method and proceeds as before.

Provided the user was acquainted with differential equations and their solutions, he could learn how to do this in approximately 20 minutes. This same user might also, in a few additional minutes, learn how to do algebraic manipulations so that he might seek analytic solutions to this differential equation.

Another feature of CAMA is the economy of its operations. Although many of the parts are not optimally coded in the sense of maximum speed of operation, all parts in CAMA are controlled in such a way that costs are minimized. The charging algorithm on the central computer depends upon many factors, two of which are: the amount of storage being used, and the length of time storage is being used. In other words, the integral of the amount of storage over time determines the major cost of computer use. If the user pauses to think, the CAMA system automatically moves all of his operations out on to disk files, and keeps only a small portion of his program in the computer to await his next command. Thus, the charges during this thinking time are very small. They might perhaps be as low as $3 for a half-hour's worth of thought. When the user becomes active again, the system responds very quickly, bringing his problem back into the virtual memory and, for most operations, acting as if it were responding immediately to his request for service.

There are three basic modes of operation of the CAMA

system: the immediate mode, the delay mode, and the batch mode. The immediate mode responds at once to the user's request. When he asks for a certain operation, the machine returns the results to him as quickly as it can produce them. The delay mode is somewhat different. If, for efficiency, the user wants to stack a series of operations and invoke them as a unit, he may specify these operations much as he would in ordinary programming, except, of course, that he is using graphical input and output. When all the operations to be invoked have been stacked, the user then causes them to be executed. He can, if he wishes, build into this stack of operations, monitoring features which allow him to interrupt or observe the processing. The third mode of operation is called batch, although it is not a batch operation in the ordinary sense. It resembles the delay mode, except that the operations are executed later, when the user is off the terminal. This is desirable in some cases because off-terminal (non-interactive) operations are cheaper than on-terminal ones.

All modes of operation permit the user to keep, if he wishes, any intermediate results on the disk files (up to the limits of his allocated storage) and use them as the basis of further operations. In mathematical processes it is frequently necessary to back up and do something a different way. If the user kept sufficient intermediate

information, this backing up process is rather easy. For example, in solving a differential equation, he may find that the method he has chosen is unstable. By merely accessing the data storage pack which contains the original specification of the differential equation, he can seek a more stable method of solutions without having to respecify the differential equation. Furthermore, by keeping appropriate pieces of infomation on disk files, he can resume operations at a later time without starting the problem over from the beginning. This capability is highly important in mathematic analysis because the student often doesn't know where he is going or how to get there, but rather is conducting experiments in a mathematical language.

In algebraic manipulation, the user cannot always specify everything he wants by simple commands. To overcome this difficulty, CAMA provides many commands derived from basic algebraic manipulations such as the commutative law, the distributive law, the associative law, and so forth. Furthermore, built upon these primitive operations is a hierarchy of more complex operations, each of which can be called individually under the control of the user at the console. Or he may define new combinations for his own use.

The algebra or calculus built into the CAMA system is not limited to ordinary algebra. Predefined matrix

algebra, physical vector algebra, and several others are also available. More complicated algebras can be defined and some of the CAMA designers are presently incorporating these into the system.

All CAMA operations are under direct user control via light-button selection.[3] Because of the small size of the display screen, only a selected group of operations can be displayed at ony one time. However, by a simple command at the teletype, the user can add or delete from the screen light buttons which specify other actions.

It is relatively easy to define a new operation and its associated light button by identifying a particular light button with a particular task (a simple command does this), and defining that task as a sequence of other tasks. The user can also define his own operations either as a combination of existing primitives, or, if he is sufficiently knowledgeable, as a combination of primitives he himself has defined.

3. HARDWARE CONFIGURATION

The University of Michigan Computing Center has an IBM 360/67 duplex computer with a four-billion-byte virtual memory consisting of about 1.5 million bytes of directly addressable core storage and the remainder on an automatically paged drum (see Figure 1). Auxiliary storage consists of disk files of 400-million bytes, a 400-million-byte data cell, and several magnetic tape units. The main computer is accessed on a timesharing basis through many ports. CAMA uses the one known as the Data Concentrator.[4] Over one hundred other remote data terminals can be connected simultaneously via telephone lines to the central computer. Accessible of course are other things such as high-speed printers, punches, card-readers, and the like. A large-bed Calcomp plotter is also available.

In a building several blocks away is the terminal computer, a Digital Equipment Corporation PDP-8 with two 103 Dataphones and one 201 Dataphone. Two teletypes are connected to the PDP-8 through two 103 dataphones, and the PDP-8 is connected to the central computer through the Data Concentrator by the 201 data link. All of these links are dialable facilities through the regular telephone lines. Associated with the terminal computer is a disk file system, a display with a light pen, a high-speed paper-tape reader/punch, a Grafacon, and a small Calcomp plotter. All of these facilities can be used by the CAMA system.

HARDWARE CONFIGURATION

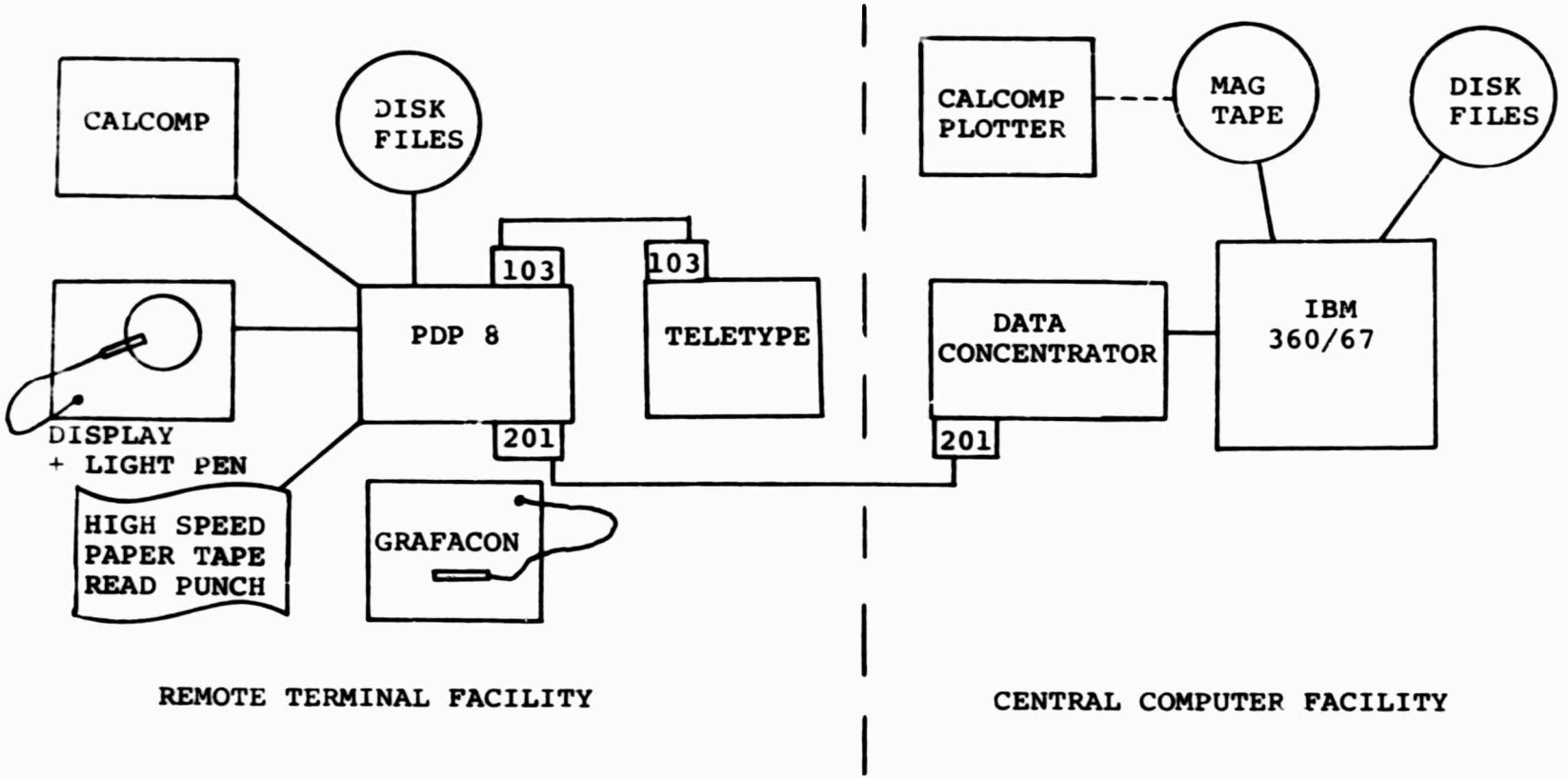


Figure 1

4. CENTRAL COMPUTER SOFTWARE

The Michigan Terminal System (MTS) is the operating system written by the Computing Center staff at the University of Michigan for the IBM 360/67 computing system.[5] This system permits the concurrent handling of a large number of jobs either in batch-processing mode or time-sharing mode. MTS is a multiprogramming system that oversees all computational activities, including control of access to the computer, management of and operations on files, translation of source-language programs, execution of object programs, and maintenance of all accounting records.

Usually MTS treats all active users alike, rendering service in round-robin fashion. However, MTS may assign priority levels to users when more than one user is competing for the same computing resource.

The user communicates with the operating system through the MTS command language, an artificial language with a grammar and vocabulary defined by the authors of MTS.

In addition to MTS with its supervisory services, the system library contains programs that, for example, allow a user to edit a line file, to debug a program using symbolic references, to obtain file statistics, to obtain a summary of his computer account statistics, and to mount magnetic tapes.

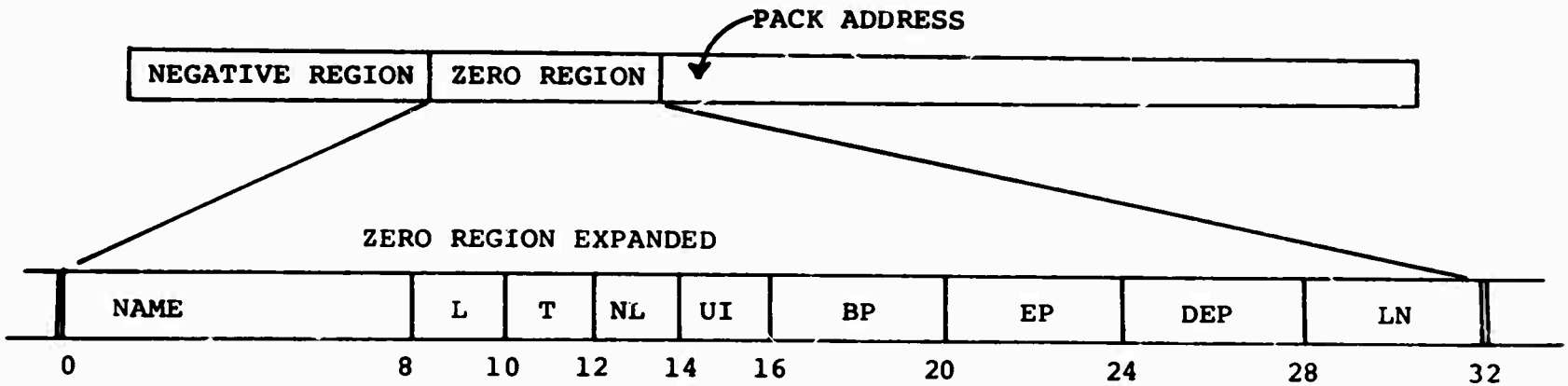
5. DATA STRUCTURES

The data structure for CAMA in the central computer was designed to meet the objective of adaptability and ease of use for non-expert users of the overall system.[6] The structure is built around data blocks that are called packs. A pack is subdivided into three regions (see Figure 2): an expandable negative region, a 32-byte zero region, and an expandable positive region.

The positive region contains the data being referenced. The zero region contains the information necessary for the handling of the packs. This includes the name of the pack, a back-pointer to a list (also stored in a pack) which references this pack, a type which is indicative of the kind of information stored in the pack, as well as a number of indicators which give the size of the pack, length of data stored, and length of negative region. The content of the negative region is not specified by the designers but is available for knowledgeable users.

The pack-handling system of CAMA consists of a group of simple linked lists which themselves are stored in packs and which are referenced by an associative language. It has been optimized in the sense that it does not use much core storage, and only needed parts of it are loaded into active storage. A disadvantage of the optimization is that locating packs may require the searching of long

DATA STORAGE PACK STRUCTURE



NAME: The eight-byte EBCDIC name of pack.

L: Length of positive region.

T: Type of data being stored.

NL: Length of negative region.

UI: Usage indicator.

BP: Back-pointer to lists referencing this data.

EP: End-pointer. Points to end of pack.

DEP: Data end-pointer. Points to last place of data.

LN: Line number of pack on disk.

Figure 2

lists. Other data systems may be linked with packs. The negative region provided for in the packs allows the user to store in the packs headers and pointers of his own choosing so that he may incorporate such systems as SLIP,[7] L^6,[8] TRAMP,[9] or the Set Theoretic Data Structure.[10]

Data packs may be created or destroyed at the will of the user. They may be expanded or contracted in size depending on the quantity of information being stored. They may be removed from active storage and stored on disk files and retrieved when needed under automatic control of the program or under the control of the user. When the user pauses in his activity to think or do other chores, all the data can be shifted automatically to disk files leaving only a very small program in control.

The packs are so structured that the data in them may be accessed from any suitably coded subroutine written in FORTRAN. This allows even the least skilled programmer to write subprograms and to use his existing subprogram written in FORTRAN, MAD/I, ALGOL, or other common languages without modification. Thus, common procedures using arrays such as matrix addition, multiplication, etc. can be drawn from existing libraries.

6. TASK QUEUE

All CAMA operations in the central computer are divided into units referred to as tasks.[6] The tasks may be defined by the user or may be part of the CAMA system. Tasks may be generated by the action of the user at the console or by the system when it detects an error or when some internal activity needs attention. They are stacked on a queue and are processed on a first-on first-off basis, unless priorities are requested. Priorities are given when data is being transmitted from the terminal, when errors occur in processing such as data overflow in packs, when information is being processed, or when requested by a select group of commands. After the priority items are processed, the tasking system returns to the first-on, first-off basis of processing.

At the bottom of the queue, a series of tasks constantly survey data packs and subprograms to see if they may be discharged. Thus, during the dwell times, data management may take place, whereas when the user is most active his needs are serviced as quickly as possible.

7. DISPLAY FILE PROCEDURES

One part of the CAMA system is a group of procedures known as display file (or DF) routines.[3] These allow the user at the graphics terminal to assemble display files in the central computer, transmit them to the terminal computer, load them into the storage, and control the displaying of the display files. A display file may consist of line drawings, descriptions of characters and symbols, or calls on other display files. Thus any picture or series of symbols generated in the central computer can be easily displayed.

A particular feature of the terminal computer allows display files to call other display files as subroutines. For example, if a pattern described by a display file is to be displayed at more than one location on the display screen, only one copy of that pattern need be defined with as many references to it as necessary or desired in other display files. In particular, this feature allows the user to complement his set of characters and symbols with the standard character set which has been defined for the display screen of the terminal computer.

The DF routines include light pen and Grafacon support. These routines enable either the light pen or Grafacon and then return data to the user.

8. DRAWING CAPABILITY

Within the CAMA system is an adaptation of a simple drafting language developed at the University of Michigan known as DRAWL.[11] This adaptation uses the DF routines to produce pictures on the display or hard copies on the Calcomp plotters at either the terminal or at a central computer facility. DRAWL can describe any three-dimensional object by assembling objects into larger objects, and then rotate that object or project it onto any plane. Perspective views can also be obtained.

The principal use of DRAWL in CAMA is for the drawing of curves which are the results of calculations, producing axes for graphs, labeling of graphs, and producing bar charts. At present, there is no provision for the removal of hidden lines, but we plan to include this option at a later date.

9. DYNAMIC LOADER

The CAMA system presently incorporates approximately 600 subprograms for the central computer, exclusive of those that the user may wish to add, but only a small portion of them need be loaded at one time. Because it would be impossible and prohibitively expensive to have all 600 in virtual memory at all times, we have provided for the dynamic loading or unloading of subprograms at the will of the user or under program control.[12] An infrequently used program may be called in from disk storage when needed and released when not needed. On the other hand, frequently used programs may be loaded and kept in virtual memory as long as desired.

10. TRANSLATORS

The interpretation of the symbolic patterns created in the terminal computer and stored in the central computer is a matter for the user to decide. He may create his own translator to interpret patterns, or he may use a translation already defined.[13] Present translations include some so simple as to take a line of symbols and produce a line of FORTRAN code; others perform numerical integrations, or interpret expressions involving matrices or other algebraic entities. Some take an ordinary differential equation and produce a program for its solution by means of one or more numerical processes.

The existing translators were constructed with the aid of several processors which the knowledgeable user may employ to construct his own translators. These include a lexical scanner, a generalized macro facility, and a table-driven parser.

The macro facility allows the user to define a "language" of his own in terms of already existing languages such as FORTRAN, MAD/I, PL/I, or others, He may, in fact, define a new language in terms of another one that he has previously defined. The macro facility allows the production of as many lines of output code as are desired from a single line of input code. There is a limited but most useful context-interpreter in the macro-producer

which allows modifications of the productions on the basis of the modes of variables, counts, operations indicated, and other parts of the input strings.

The lexical scanner delimits the syntactic units of any line of code and sets up appropriate tables. The scanner, of course, depends on user-defined tables which define operators, delimiters, etc.

The parser (currently a simple precedence parser) takes the delimited input strings and produces sets of triples. The precedences are set in a table which can be defined by the user if he so desires. For the several predefined languages, precedence tables already are available. The user can change precedences and the external representation of operators. As an example, he might wish to name a variable "*" and denote the multiplication operation by M. Of course, the user may reduce readability in his "language," but CAMA does allow him to make such choices.

The triples which are the result of the parsing operation are sent through a macro processor, which in turn produces code in the base language. Sometimes several passes through a parser and macro processor are necessary to produce such code.

The base language may be a language such as FORTRAN, which requires the execution of a compiler to produce machine code. However, for the immediate mode of operation,

the base language is a collection of special calls within the CAMA system which can, in effect, call any subroutine in the library and cause it to operate on any data pack or group of data packs. All predefined operations can be called dynamically. With these special calls as a base langauge, operations can be performed immediately as they would be in an interpretive language. This is not a very efficient process, however, and is not in general used in the delay mode of operation. Instead, in the delay mode, a compiler or an assembler is called in to handle the base language translation. Future productions will be in machine code.

Although still in a formative state, the powerful MAD/I compiler at the University of Michigan has made the coding of CAMA very much easier.[15] The translators and much of the future development depend on this compiler.

11. ALGEBRAIC MANIPULATION

The graphical, interactive nature of CAMA makes algebraic manipulations more practical than in older systems.[16][17][18] There is distinct advantage in being able to see mathematical symbols as they usually appear in the mathematical literature as opposed to seeing them as rather obscure mnenomics. As an idea is easily lost after a delay of 15 seconds,[19] the immediate viewing of the results of an algebraic manipulation is far superior.

At present, the CAMA system has only a limited algebraic capability, but is growing as it is used. All the algebraic operations are based on a group of primitive operations. These include identifying an entity, moving an entity, substituting an expression, applying the associative law of addition or multiplication or the inverse, applying the commutative law of addition or multiplication, applying the distributive law, and the combining of like adjacent terms. Upon these primitive operations more complex operations are built. For example, the gathering of like terms from a long expression is accomplished by identifying one term as an entity, searching for identical entities, moving the entity to an adjacent position, and combining terms. The inverse operation is also available and is sometimes useful.

Many operations are currently defined for one-dimensional expressions, that is, expressions in which only the X positions of the symbols are significant. However, a limited but growing set of two-dimensional operations is also available. There are many problems associated with the ambiguous interpretation of two-dimensional expressions which have not been entirely resolved. Many of these problems are easily bypassed, however, because the user can watch the progress of the manipulation and interject the correct interpretation if the programmed interpretation is incorrect.

Inasmuch as the algebraic operations are defined in terms of primitives, and since any user can include or delete any operation at will, the ultimate capability in terms of different algebras is great. For example, it was relatively easy to include many operations of matrix algebra and calculus and physical vector algebra and calculus in this system. Tensor algebra and calculus is now being included. Although we have not pursued it to any extent, it seems possible to define algebras to manipulate physical entities such as machine parts, structural elements, or picture elements.

12. TERMINAL PROCEDURES

All the activities in the terminal computer are imbedded in an operating system known as RAMP.[20][21] This is a multiprogramming system which takes care of all such services as processing interrupts, transmitting data to and from the data phones, structuring display files, interpreting light-pen hits and Grafacon interrupts. It handles the storage and retrieval of information from the local disk files, prepares data for the local Calcomp plotter, and in general handles all the management of buffers and subprograms within the terminal computer.

Several procedures for the terminal computer are specifically designed for CAMA.[22] One such procedure allows a user to define any symbols he chooses and store them for future use in the central computer's data packs. Thus he can create any of the commonly used mathematical symbols such as partial derivative symbols, integral symbols, script letters, Greek letters, or, for that matter, any pattern. Associated with each symbol are a number of attributes, including a primary reference point which identifies the X and Y coordinates of the symbol when used in an expression, a name, and type. The attributes also include a group of secondary reference points which can serve a number of roles including points of attachment for other symbols and for connectors in such things

as electrical circuits. At present, approximately two-hundred commonly used symbols have been defined and stored in appropriate packs in the central computer.

The user may select a subset of symbols from the set of predefined symbols and/or add to that set any that he defines himself. This set becomes a menu from which he can construct the mathematical expressions or other symbolic patterns. This procedure allows the user to select the symbols by a light pen, or through the Grafacon, and locate them in whatever pattern he feels is meaningful. This procedure allows him to move the symbols, edit the pattern, or add and delete symbols. When the pattern satisfies him, he transmits it to the central computer. The information transmitted includes the x and y coordinates of the primary reference point of the symbols as well as the name, type, size, and other pertinent attributes. This information is stored in a pack named by the user. Then, by the same procedure, he may construct other expressions or patterns until he has a complete set for his purposes.

ACKNOWLEDGMENTS

The author wishes to give special thanks to Mr. W.S. Gerstenberger for his continuing assistance and support, without which this work would have been impossible, and to Mssrs. Robert W. Taylor and Richard Johnston for their most significant contributions to the CAMA system. Mrs. Jane Bisgrove, Mrs. Suzanne D. Goodrich, and Mr. Larry Julyk, although not listed as authors, deserve most of the credit for the ideas and their implementation described in this report. The author also wishes to thank the many people at the University's Computing Center and at the Concomp Project who gave invaluable assistance.

REFERENCES

1. Culler, G.J., Fried, B.D., "The TRW Two-Station On-Line Scientific Computer: General Description," Computer Augmentation of Human Reasoning, Spartan Books, Washington, D.C., 1965.

2. Kierer, Melvin, and May, Jack, "Two-Dimensional Programming," Proceedings 1965 FJCC, p. 63.

3. Cocanower, Alfred B., The DF Routines User's Guide, Memorandum 23, Concomp Project, University of Michigan, Ann Arbor, May 1969.

4. Mills, D.L., The Data Concentrator, Technical Report 8, Concomp Project, University of Michigan, Ann Arbor, May 1968, 113 pp.

5. MTS: Michigan Terminal System, 2nd ed., Computing Center, University of Michigan, Ann Arbor, December 1967, 2 vols.

6. Julyk, Larry, and Wolf, Louis W., The CAMA Data Structure, Memorandum 29, Concomp Project, University of Michigan, Ann Arbor, August 1970.

7. Weizenbaum, J., "Symmetrical List Processor," Communications ACM, September 1963, pp. 524-536.

8. Knowlton, K.C., "A Programmer's Description of L^6," Communications ACM, August 1966, pp. 616-625.

9. Ash, W., and Sibley, E.H., TRAMP: A Relational Memory with an Associative Base, Technical Report 5, Concomp Project, University of Michigan, Ann Arbor, May 1968, 80 pp.

10. Childs, D.L., Description of a Set-Theoretic Data Structure, Technical Report 3, Concomp Project, University of Michigan, Ann Arbor, March 1968, 27 pp.

11. Herzog, B., DRAWL 70: A Computer Graphics Language, Technical Report 30, Concomp Project, University of Michigan, Ann Arbor, August 1970.

12. Julyk, L.J., The CAMA Operating System, Memorandum 30, Concomp Project, University of Michigan, Ann Arbor, August 1970.

13. Julyk, L.J., Dingwall T., and Wolf, L.W., The CAMA Macro Processor, Memorandum 35, Concomp Project, University of Michigan, Ann Arbor, August 1970.

14. Dingwall, T., Julyk, L.J., and Wolf, L.W., The CAMA Interpreter, Memorandum 36, Concomp Project, University of Michigan, Ann Arbor, August 1970.

15. Bolas, B., Srodawa, R., and Springer, A., The MAD/I Manual, Technical Report 32, Concomp Project, University of Michigan, Ann Arbor, August 1970, 198 pp.

16. Bond, E., et al. "FORMAC - An Experimental FORmula MAnipulation Compiler," Proceedings of the ACM National Conference, August 1964.

17. Engelman, C., "MATHLAB, A Program for On-Line Machine Assistance in Symbolic Computations," Proceedings FJCC 1965, pp. 413-421.

18. Sibley, E.H., The Engineering Assistant: Design of A Symbol Manipulation System, Technical Report 2, Concomp Project, University of Michigan, Ann Arbor, August 1967, 31 pp.

19. Miller, Robert B., "Response Time in Man-Computer Conversational Transactions," Proceedings FJCC 1968, p. 267.

20. Mills, D., RAMP: A PDP-8 Multiprogramming System for Real-Time Device Control, Concomp Project, University of Michigan, Ann Arbor, May 1967, 24 pp.

21. Gerstenberger, W.S., and Taylor, R.W., Graphics RAMP User's Guide, Memorandum 34, Concomp Project, University of Michigan, Ann Arbor, August 1970.

22. Goodrich, Mrs. S., CAMA: Define-Problem Command, Memorandum 28, Concomp Project, University of Michigan, Ann Arbor, June 1970, 31 pp.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D
(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) UNIVERSITY OF MICHIGAN CONCOMP PROJECT	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

CAMA (COMPUTER-AIDED MATHEMATICAL ANALYSIS): A GENERAL DESCRIPTION

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Memorandum 33

5. AUTHOR(S) (First name, middle initial, last name)

LOUIS W. WOLF

6. REPORT DATE October 1970	7a. TOTAL NO. OF PAGES 28	7b. NO. OF REFS 22
8a. CONTRACT OR GRANT NO. DA-49-083 OSA-3050 b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) Memorandum 33 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	

10. DISTRIBUTION STATEMENT

Qualified requesters may obtain copies of this report from DDC.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Advanced Research Projects Agency

13. ABSTRACT

This paper describes CAMA (Computer-Aided Mathematical Analysis), which is a system that attempts to handle all conceivable mathematical operations, from simple numeric procedures to complex algebraic or set-theoretic manipulations. CAMA operates in a timesharing environment with a large central computer and a remote terminal computer with graphical devices.

Some of the features of CAMA are: the input and output of mathematical symbols in their usual mathematical representation; the ability to manipulate these symbols according to predefined algebraic laws; the definition by the user of other algebraic laws; the definition of symbols with user-prescribed meanings; the automatic translation of expressions or equations into any of several languages; and numerical calculations. The user can invoke any of these tasks through the graphical terminal.

DD FORM 1 NOV 65 1473

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
graphical languages symbol manipulation data structure string handling						